OpenPOWER™

The Little-Endian OpenPOWER Software Development Environment

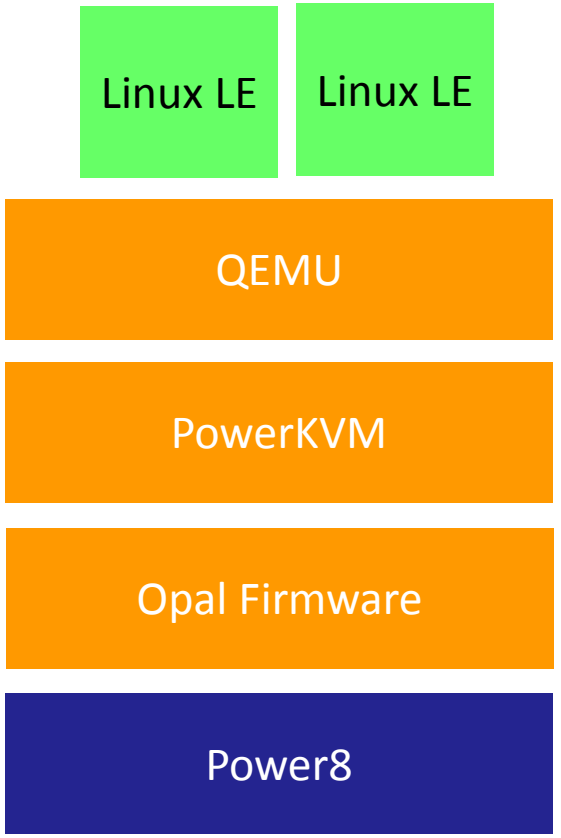
Dr. Michael Gschwind

Senior Technical Staff Member & Senior Manager

IBM Power Systems

OpenPOWER ecosystem

- Enable rich ecosystem of hardware vendors
  - Standardized hardware interfaces
  - ⇨ Common, open firmware interfaces
- Open source system software stack
  - Data center operators rely on tuning SW stack
  - Enable server ODM vendors to create offerings
  - ⇨ Operating environment built on Linux and KVM



A New OpenPOWER Linux Environment

- OpenPOWER is not traditional Power Linux with a new name
  - Significant discontinuity and fresh start
  - ⇨ new environment “ppc64le”
    - Firmware, Hypervisor, data layout, source code, ABI, APIs

- What changes for application developers?
  - Byte order
  - New ABI
  - Vector programming API

The New Byte Order

- ⇨ Little-endian data format and programming interfaces

- Simplify porting of applications previously locked to Intel x86
  - Large-scale data center applications
  - Application source code dependences

- Access data repositories storing binary data written by Intel x86
  - In-storage data base formats
  - Data sharing with mobile devices

- Simplify data sharing with I/O devices originally from LE ecosystems`
  - Easily exploit I/O and accelerators designed for PCs and mobile devices

The New OpenPOWER Application Binary Interface (ABI)

- Optimizations driven by hardware and software evolution
  - Align with the broader ecosystem
  - Create hardware optimization opportunities and synergies
- Optimize for modern code patterns
  - More classes, abstraction
  - Shorter function lengths
  - More indirect calls
- New capabilities as delta over starting point: PPC64 / AIX ABI
  - Established, tested production code
  - Commonality and maintenance across LE, BE and AIX where feasible
  - Minimum disruption for tooling: GCC, XL, Java, LLVM, libffi, PyPy, …

The new ABI

- Application development: no change
  - for applications written in Fortran, Pascal, C, C++, C#, Java, Python, Ruby, Spark
  - minimal change (0-4 lines) for assembly programs
- Align with other Linux environments to simplify migration
- Simpler and shorter code
- Improved performance and ease of adoption
  - During initial LE Linux bringup, 40,000 packages ported over short period

ABI Improvements: simplify and accelerate

- Global data management with the Global Offset Table
  - Initialize pointer to GOT without functions descriptors
  - Optimize GOT pointer update on cross-module calls
  - Expand addressing range with “Medium Code Model”
  - Exploit Displacement Fusion and avoid GOT overflow code

- Pass more registers in their native registers
  - Reduce abstraction penalty (“same performance as builtin types”)
  - OO languages wrap types in abstract class
  - Previously classes handled differently from builtin types
  - ELFv2 passes up to eight class members in registers
  - Return function results in same register(s) as first input parameter

Medium Code Model

- "Medium code model" addresses growing application size
  - Expand GOT data dictionary to up to 4GB
- Avoid expensive GOT overflow > 64KB (8k variables) per module
  - Size originally set to POWER ISA offset size (16b)
- Enable applications with up to 500M variables per module
  - "Beyond RISC" using Displacement Fusion in Power8

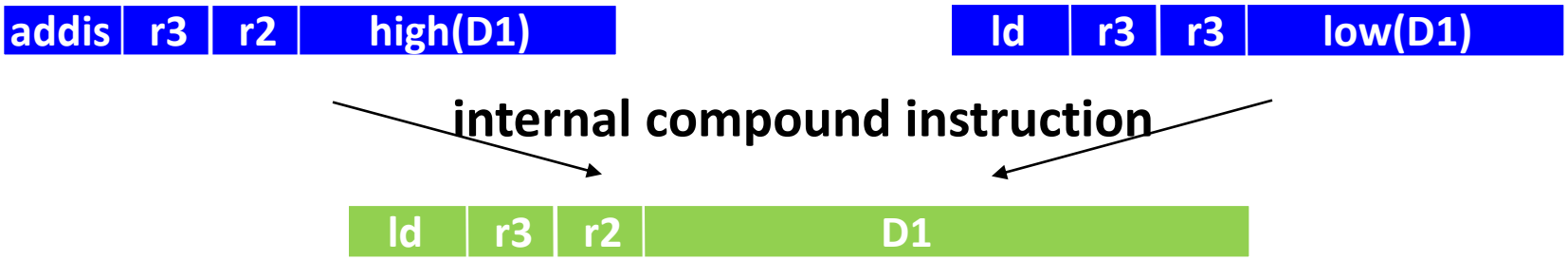
Beyond RISC: Displacement Fusion

- Combine multiple instructions into single internal instruction
- Compound instruction executes as a single hardware operation
- Increased addressing range with RISC fixed-width instr. advantage

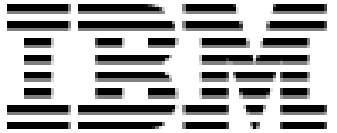
```
addis  r3=r2, D1@ha
ld     r3=r3, D1@l
```

Displacement fusion

```
ld     r3= r2, D1
```



Example OO method with abstract data types



```
                          PDEF      _vec8_mulfp
199|                                      PROC      #retvalptr_49,left,right,gr3,gr5-gr10
  0| 0031A0 stdu      F821FF31    1      ST8U      gr1,#stack(gr1,-208)=gr1
206| 0031A4 addi      380000C0    1      LI        gr0=192
203| 0031A8 addi      38810130    1      LA        gr4=right(gr1,304)
  0| 0031AC std       F8C10118    1      ST8       left(gr1,280)=gr6
  0| 0031B0 std       F8A10110    1      ST8       left(gr1,272)=gr5
  0| 0031B4 std       F8E10120    1      ST8       left(gr1,288)=gr7
  0| 0031B8 std       F9010128    1      ST8       left(gr1,296)=gr8
  0| 0031BC std       F9210130    1      ST8       right(gr1,304)=gr9
202| 0031C0 addi      38A10110    1      LA        gr5=left(gr1,272)
  0| 0031C4 std       F9410138    1      ST8       right(gr1,312)=gr10
203| 0031C8 lxvd2x    7C202698    1      VLQD      vs1=right(gr4,0)
202| 0031CC lxvd2x    7C002E98    1      VLQD      vs0=left(gr5,0)
203| 0031D0 addi      38C40010    1      AI        gr6=gr4,16
202| 0031D4 addi      38850010    1      AI        gr4=gr5,16
205| 0031D8 xvmulsp   F0400A80    1      VFM       vs2=vs0,vs1,fcr
203| 0031DC lxvd2x    7C203698    1      VLQD      vs1=right(gr6,0)
202| 0031E0 lxvd2x    7C002698    1      VLQD      vs0=left(gr4,0)
206| 0031E4 xvmulsp   F0000A80    1      VFM       vs0=vs0,vs1,fcr
205| 0031E8 addi      388000B0    1      LI        gr4=176
205| 0031EC stxvd2x   7C412798    1      VSTQD     result.m128_0(gr1,gr4,0)=vs2
  0| 0031F0 ori       60420000    1      XNOP
206| 0031F4 stxvd2x   7C010798    1      VSTQD     result.m128_1(gr1,gr0,0)=vs0
207| 0031F8 ld        E80100B0    1      L8        gr0=result(gr1,176)
207| 0031FC std       F8030000    2      ST8       #retval_49(gr3,0)=gr0
207| 003200 ld        E88100B8    1      L8        gr4=result(gr1,184)
207| 003204 std       F8830008    2      ST8       #retval_49(gr3,8)=gr4
  0| 003208 ori       60420000    1      XNOP
207| 00320C ld        E80100C0    1      L8        gr0=result(gr1,192)
207| 003210 ld        E88100C8    1      L8        gr4=result(gr1,200)
208| 003214 addi      382100D0    1      AI        gr1=gr1,208
207| 003218 std       F8030010    1      ST8       #retval_49(gr3,16)=gr0
207| 00321C std       F8830018    1      ST8       #retval_49(gr3,24)=gr4
208| 003220 bclr      4E800020    0      BA        lr
```

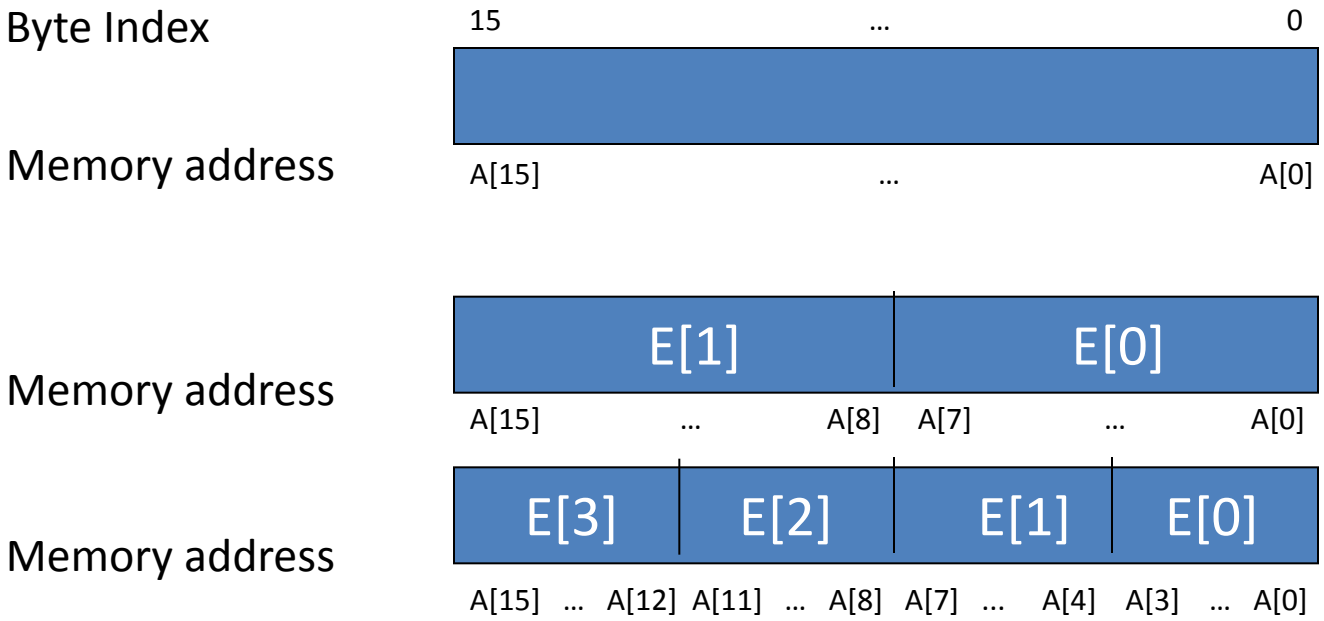
- Work instructions in green
- Overhead due to passing abstract data types via memory in red

Courtesy of Ian McIntosh, IBM Toronto

OpenPOWER™

The Little-Endian Vector API

- Vector API builds on OpenPOWER Little-Endian Data Model
  - Focus on programmability – consistent little-endian view
  - Focus on ease of sharing code with other little-endian ecosystems



Little-Endian Vector API Implementation

- Common Vector Programming API across GCC, LLVM and XL
  - Vector builtins as operators
  - Enable compilers to optimize expressions with vector operators

- Vector API code models are programmer abstractions
  - Common LE/BE compiler backend...
  - … recognizes and optimizes the different conventions

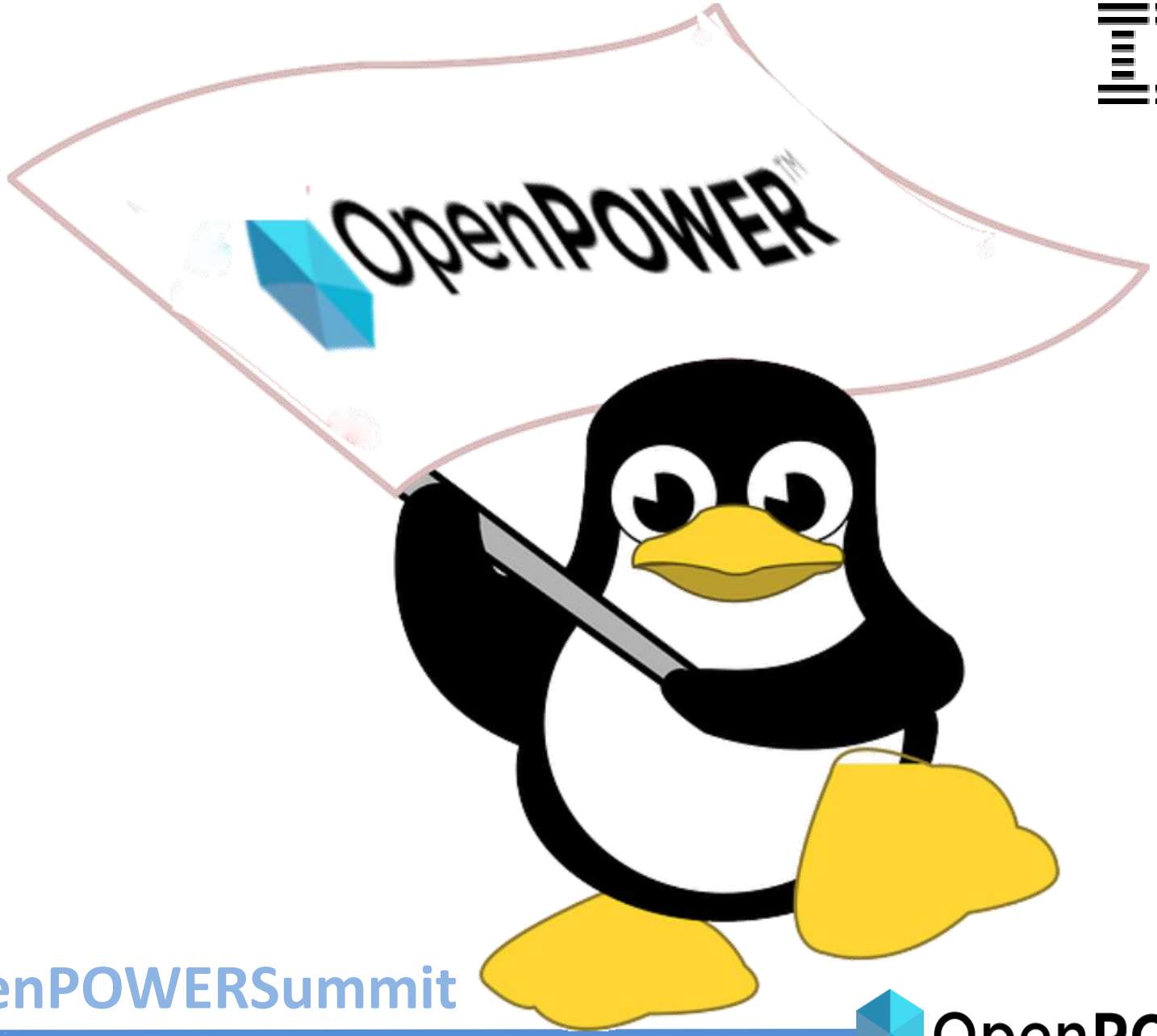
$$f^{-1}(s(f(x), f(y))) \Leftrightarrow s(x,y)$$

Technical details:

- "Supporting Vector Programming on a Bi-Endian Processor Architecture", LLVM 2014
- "Bridging endian-dependent SIMD vector representation with compiler optimization", PPoPP Workshop on Programming Models for SIMD/Vector Processing 2015

OpenPOWER Environment available now

- Collaborative innovation already changing industry
  - Major data center stakeholders joining OpenPOWER
  - Little-endian Linux on Power available from three major distros
  - → Over 40000 open source packages and ISV applications ported
- Redefined software stack: Firmware, Hypervisors, OS, Applications
  - Little-endian data model for simplified application porting
  - New ABIs and APIs support developers in exploiting platform
- New OpenPOWER environment enables
  - Ease-of-use and out-of-box performance
  - Exploitation of new Power8 hardware features



#OpenPOWERSummit

OpenPOWER™

Special Notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Special Notices (cont.)